-448

FACTORS THAT AFFECT THE COST OF COMPUTER PROGRAMMING

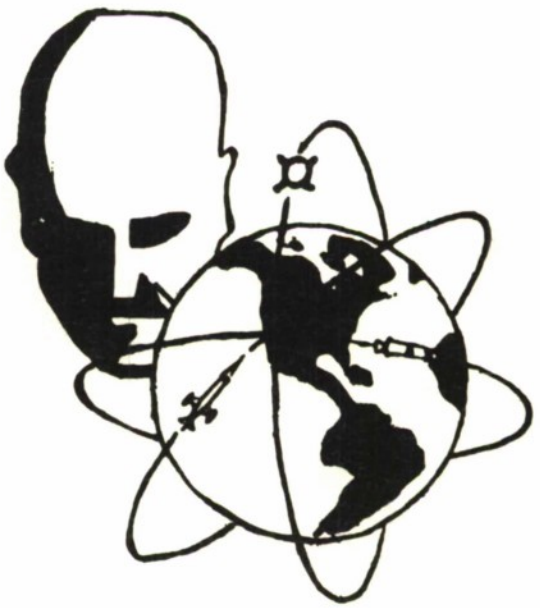
VOLUME I

TECHNICAL DOCUMENTARY REPORT NO. ESD-TDR-64-448

JULY 1964

L. Farr
B. Nanus

DIRECTORATE OF COMPUTERS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts



(Prepared under Contract No. AF 19 (628)-1648 by the System Development Corp., Santa Monica, California, 90406.)

DDC AVAILABILITY NOTICES

Qualified requesters may obtain copies from Defense Documentation Center (DDC). Orders will be expedited if placed through the librarian or other person designated to request documents from DDC.

Copies available at Office of Technical Services, Department of Commerce.

ESD-TDR-64-448

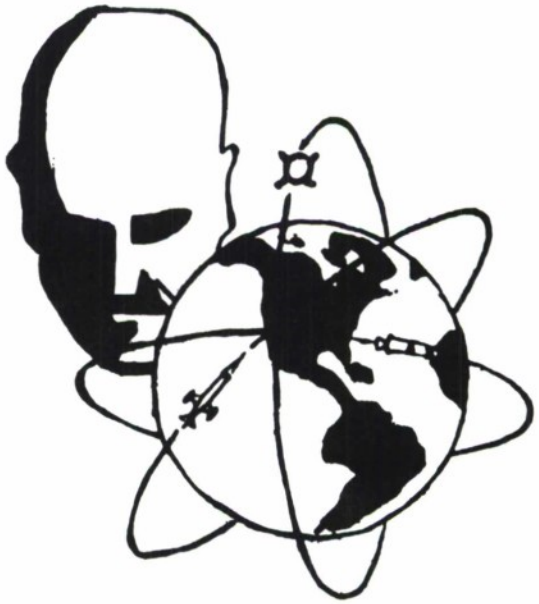
FACTORS THAT AFFECT THE COST OF COMPUTER PROGRAMMING

TECHNICAL DOCUMENTARY REPORT NO. ESD-TDR-64-448

JULY 1964

L. Farr
B. Nanus

DIRECTORATE OF COMPUTERS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts



(Prepared under Contract No. AF 19 (628)-1648 by the System Development Corp., Santa Monica, California, 90406.)

FOREWORD

This document is the first of two Technical Documentary Reports prepared for the Air Force Electronic Systems Division as part of a project to develop better techniques for estimating the costs of computer programming. It originally appeared as a System Development Corporation document, TM-1447/000/01.

The material contained in this report served as the basis for shorter papers presented by the authors at two professional meetings: SDC document SP-1372/000/01, Cost Aspects of Computer Programming for Command and Control, at the Winter Military Electronics Conference in Los Angeles in February 1964 and SP-1376/000/01, Some Cost Contributors to Large-Scale Programs, at the Spring Joint Computer Conference in Washington, D. C. in April 1964. The authors wish to acknowledge the assistance of their associates at the System Development Corporation who contributed many ideas and suggestions in the course of this work, particularly V. LaBolle and N. E. Willmorth. They also appreciate the early support and encouragement of this work by the Advanced Research Projects Agency of the Directorate of Defense Research and Engineering.

Leonard Farr

Burt Nanus

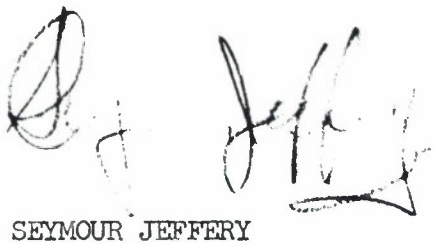
FACTORS THAT AFFECT THE COST OF COMPUTER PROGRAMMING

ABSTRACT

Although accurate estimation of computer programming costs is an important prerequisite for effective programming management, such estimates have historically been very unreliable. Some of the underlying causes of this problem are discussed, and about fifty factors that appear to contribute to the cost of computer programs are identified. Data concerning the effects of a few of these factors upon cost are presented by way of illustration. Recommendations are made for more detailed cost collection, cost analysis, and experimentation.

REVIEW AND APPROVAL

This technical documentary report has been reviewed and is approved.



SEYMOUR JEFFERY
Major, USAF
PROJECT OFFICIAL

KEY WORD LIST

1. PROGRAMMING (COMPUTERS)
2. COSTS
3. EXPERIMENTAL DATA
4. ANALYSIS

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

I. INTRODUCTION

One of the most important requirements for management planning is an accurate estimate of the resources required to complete the project. In programming management, the two principal resources to be estimated, scheduled, and controlled are labor--measured in man months, and computer use--measured in computer hours. Together, these two resources may be considered the variable cost of producing the program. Reliable methods to estimate them are not available. Historically, costs of programming have been estimated very poorly; in fact, examples of budget overruns exceeding 100 percent have been reported. Because cost estimation is a significant first step toward more effective allocation of resources by programming management; because the costs of programs may be a significant portion of the total costs of command and control systems; and because the estimates have been little better than guesswork to date, research toward better procedures for estimating is needed.

Development of a list of factors that contribute to cost is a logical first step toward this goal. Such a list may also serve as a basis for recommendations concerning the types and kinds of data that should be collected, and as a guide to more efficient resource planning, control, and expenditure of funds during the implementation of a computer programming effort.

A. PROBLEMS IN DETERMINING COST FACTORS

During this initial effort, members of the Computer Program Implementation Process (CPIP) project have been faced with difficulties that deterred effective research in the cost area. These problems include the following:

1. Lack of Agreement on Terminology

The definitions of many of the terms used in computer programming are not universally acknowledged. While there are many programming glossaries, such as those prepared by the Association for Computing Machinery (ACM) and the Bureau of the Budget, these are not in very wide use and virtually every programming organization develops its own set of working definitions to suit its own needs. Even if these glossaries were accepted, they presently lack definitions of the programming process, the products and the personnel. Specifically, there is a need for definitions that permit easy comparison of programming efforts.

2. Poor Definition of Product Quality

Little attention has been given to attributes that characterize the nature or the quality of a computer program as a product. Those efforts that have been initiated have seen little success, particularly in definitions of quantitative measures of quality and

performance. This lack of standard measures hampers reliable comparison of costs among various program systems. For example, programmers use the terms "flexibility," "tightness of coding," and "maintainability," but there seem to be no generally agreed upon criteria for comparing similar programs on the basis of these attributes.

3. Poor Quality and Paucity of Cost Data

Present cost collection methods seem to be used primarily for accounting purposes and not for planning or control. For example, contract costs are collected according to organizational units rather than product or function to be performed. The effect of this practice is that many of the costs that are collected by various organizations are not comparable. Also, these data are usually not well defined and, therefore, are not reliable for analytic purposes.

4. Dynamic Nature of the Field

Automatic data processing is a newly developed and constantly changing technology. On the other hand, information-processing personnel tend to "reinvent" and to use acronyms as labels; thus there is difficulty in identifying what is really new. As a result, in the absence of an analysis that might penetrate the "acronym barrier," any cost analysis based upon experience data may have to be confined to a small sample size which, in turn, increases the margin of uncertainty in any predicting methods that result--and, hence, in the cost estimates yielded by them. Despite this drawback, continued analysis can significantly reduce the uncertainty in today's estimates.

5. Nonquantitative Nature of Some Factors

Many of the factors that appear to affect costs of computer programs are qualitative. In some cases, it is possible to predict at least the direction that cost will be affected by an increase in a given factor--for example, one would expect that the more experience the contracting agency has with the particular type of program involved, the less it will cost to perform the contract (all other factors being equal). In other cases, it is not at all clear whether an increase in the given factor will increase or decrease cost. Also, we would expect some factors to be nonmonotonic or to increase cost in one area while decreasing it in another. In this respect, the list of factors may also serve as a source of ideas for future research in program costing.

In the programming discipline, these limitations apply equally well to development of management aids other than cost-estimating relationships. These problems are being more commonly recognized and subjected to study,

and remedial measures are being formulated. Meanwhile, despite the deterrents listed above, analysis to improve cost estimation is continuing.

B. CURRENT METHOD OF COMPUTER PROGRAM COSTING

In general, computer program costing, as it is performed today, can be outlined as follows:

(1) Similarities are determined between programs required and programs known by the estimator to exist.

(2) On the basis of this analysis of experience, the size of the new program (number of instructions) or subprograms in a program system is estimated.

(3) This estimate of number of program instructions provides an intermediate parameter with which to estimate man months and computer hours. Some "rules of thumb" and conversion factors have evolved to effect this step (see page 13). Some of these rules of thumb will be discussed in more detail below. (Unfortunately, few managers or experts expose their rules by documenting them.)

(4) Man months and computer hours may then be converted to dollars by multiplying by some average rates.

(5) Finally, funds for computers, equipment, office facilities, travel, overhead and general and administrative costs are added, for the grand total.

Consideration of the factors detailed in this paper would allow a manager to perform Steps (1), (2) and (3) somewhat more completely and systematically. Information concerning many of the factors in the list may be determined at the contract proposal phase, while information on others may not be known until design is underway.

C. SCOPE OF THE DOCUMENT

The cost factors listed below are based upon experience. Specifically, the factors represent answers by managers to questions such as, "Why did you overrun your budget?" or, "Why did your program cost more or take longer to develop than another program that appears to be similar?" We have listed as many of these cost factors (answers) as we could identify and collect. We mean by cost factors those variables that will affect the expenditure of either man months or computer hours. The costs, as measured in man months and computer hours, are not independent variables (see Figure 1), and many of the factors listed influence expenditure of both resources. Some factors affect only labor or only computer use.

In some cases, a short discussion of the factor provides some ideas on (a) the correlation of cost with the factor, (b) some rule of thumb concerning the cost contributing effect of the factor, or (c) whether any information can be collected about the effect of the factor.

In some cases, some experience data are shown, as in Figure 1. These data should be regarded primarily as evidence that the process of programming is susceptible to analysis. Although comparison with other data provides insight to the reader, we do not recommend use of the data for estimation purposes because the data were collected without benefit of rigorous definitions and standards. Also, as pointed out above, they may not be reliable. The heavy blacklines are "eye-fits" in all cases and are not the result of statistical analysis.

Since many of the factors are interdependent, we would expect to find high correlation among some of them in any statistical analysis. Therefore, one difficulty we encountered was how to classify the cost factors without excessive gaps or overlaps. For example, factors could have been grouped into categories by work phase, such as program design or test; by management activity, such as planning or evaluation; by units of cost measurement, such as man months or dollars; or by the classic accounting method of direct and indirect costs. These schemes all seem to cause difficulties because of ambiguities and extensive overlap. The classification scheme chosen for this paper was selected because it includes all of the factors with a minimum of overlap (see Table I), and the categories are appropriate for initial planning by managers. In each category, the factors are underlined and followed by a discussion. In future work we will identify the dependencies and hierarchies of factors. Such analysis depends upon hard data. Therefore, the paper concludes with (1) recommendations for additional work in which cost factors can be further detailed and studied, and (2) some recommendations for specific cost data to be collected.

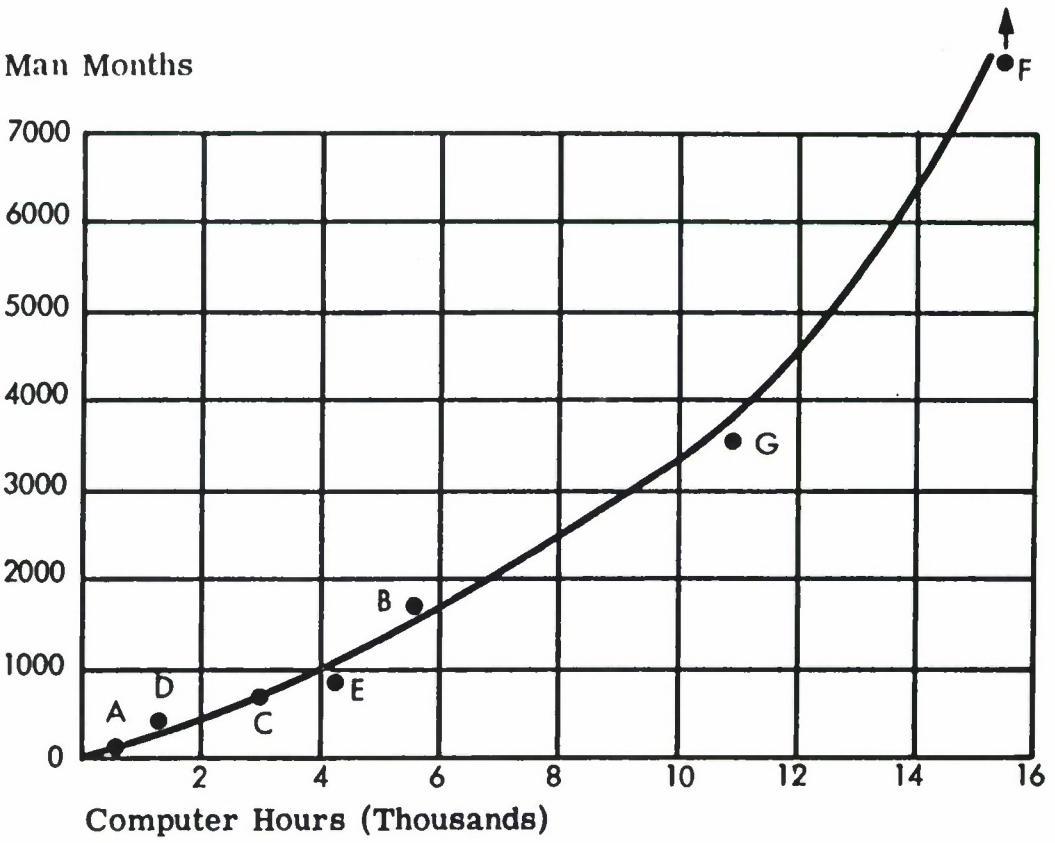


Figure 1. Man Months Versus Computer Hours

II. COST FACTORS OF COMPUTER PROGRAMMING

A. THE JOB TO BE DONE

Factors that arise from the requirements of the system to be designed and produced are included in the following categories concerned with operational and program design.

1. Operational Requirements and Design

> The Operational Requirements and Design category includes cost factors associated with the operating characteristics of the system for which the program is being developed (including the availability of such information).

The computer program should be considered a component of the information-processing subsystem which is itself a component of a larger system, e.g., a command and control system. Therefore, the analysis and design activity for computer programming cannot easily be separated from the same activity for the over-all system design. If an attempt is made to isolate the computer program as an independent component, many problems arise, especially when the intended design and operation of the computer program is dependent upon other components or subsystems and information-processing policies or procedures that constitute the larger system in which the computer program will be embedded. (This discussion is oriented toward development of operational programs for a military system, but the same considerations apply to the development of computer programs for business, or of support or utility programs.)

The analysis of information-processing subsystem requirements aims to provide detailed specification of the performance requirements of this subsystem. Ideally, the requirements analysis and specification of resulting performance requirements are provided by the customer or user of the larger system. In fact, it is often necessary for the programming contractor to assist in the determination of these requirements, because the customer may have difficulty in identifying them, and/or the programming contractor may have difficulty in translating the language of the user into the language meaningful to his own discipline (i.e., computer programming).

The aim of the operational design activity that follows analysis of information-processing requirements is to specify how the needs indicated in the requirements analysis will be satisfied by the information-processing subsystem. In other words, the statement of performance requirements must be translated into an operating system description and operational design specifications that

TABLE I--COST FACTOR CLASSIFICATION SCHEME

Logical Grouping	Category Name	Category Definition
THE JOB TO BE DONE	1. Operational Requirements and Design	Includes cost factors associated with the operating characteristics of the system for which the program is being written.
	2. Program Design and Production	Includes cost factors associated with both support and operational programs as determined by the constraints imposed by personnel, hardware and operational requirements.
THE RESOURCES THAT ARE AVAILABLE	3. Data Processing Equipment	Includes cost factors associated with the hardware required to produce and test a program, including all input, output and peripheral equipment.
	4. Programming Personnel	Includes cost factors resulting from the direct labor needed to completely develop a program.
THE NATURE OF THE WORKING ENVIRONMENT	5. Management Procedures	Includes cost factors associated with the plans, policies, practices and review techniques used in the administration of all phases of program development.
	6. Development Environment	Includes cost factors resulting from relationships with external organizations, including customers and other contractors.
	7. Facilities, Services and Supplies	Includes cost factors related to supplies, physical plant, indirect labor, and overhead.

include the division of functions and tasks between men and machines, and the procedures for handling information. These procedures include the specific requirements for the computer program.

The degree to which the factors in this category affect cost centers around the question, "How well are the operational requirements of the system known?" If they are clearly known, the programming job is more straightforward and less costly. If they are not clearly known, as is usually the case, the costs increase significantly as the programming contractor attempts to clarify and detail them.

Therefore, completeness, clarity, and detail in statements of requirements tend to have a damping effect upon costs. The factors in this category start with broad considerations and proceed to specific operational design considerations. In general, when clarification and detailing are necessary, the earlier in the system development process the programming contractor can work with the user, the greater the possibility of cost savings in the total computer program development effort.

(1) Extent of innovation in the system, its components, and especially the automatic data-processing function.

The extent of innovation in the new system, or its similarity to older systems, may be a clue to estimating how clearly or easily its requirements can be stated. Similarity to other systems, to the extent it is known, would certainly seem to indicate lower costs, while innovation and new applications are clearly more expensive to design and implement. Learning costs, although usually not identified explicitly, are usually significant.

Clearly, this factor cannot be measured easily. However, one can consider schemes that will yield a number, for example, listing the system components and determining whether or not they are new to the evaluator(s) would be one measure of newness of equipment. A similar scheme can be used to measure newness of functions.

(2) Extent to which the programming designer will participate in a determination of the information-processing needs (i.e., the system and operations analysis, and the system and operational design).

It is possible for a programming designer to be awarded a contract for computer program development before, during, or after completion of the system analysis and design activities. In general,

the earlier in the development process the programming designers (programming specialists) can begin, the greater the probability of a well designed and integrated information-processing system, with fewer costly changes. The later in the development process the programming organization begins, the greater is the probability that previous analysis must be repeated (if performed at all) in the determination of the requirements for data processing.

When the programming designer is called upon to assist in the formulation of system requirements and design, a reduction in the cost of the program system results. This provides early understanding on the part of the programming designer of the operational problems faced by the using command; promotes early identification of communications channels needed; and both establishes and exercises these channels between the user and the program developer.

This factor, the extent of the program developer's participation in the requirements analysis, is qualitative. Various methods could be used to approximate a measure of the factor. For example, in development of the large Air Force command and control systems, work is being done to improve management control of the software, i.e., nonhardware effort. The control is to be effected by requiring that specific documents be developed during the early phases (e.g., the Conceptual and Project Definition Phases in DOD language). To measure extent of program developer participation, one could simply count the documents to which some effort was devoted by the programming specialists and compute the percentage of the total documents that characterize the analysis.

(3) Number, size, frequency, and timing of system design changes.

The degree of confidence and assurance the customer displays in presenting his statement of requirements may be a clue to the number of design changes to be expected in the course of system development. Because of changes in the system environment or improved understanding of it, information-processing system changes may occur in the functions, objectives or components of the system. Although evolving systems are characterized by change, costs do increase with an increase in the number of changes. The cost of introducing any specific change depends upon the degree to which change has been anticipated in both design and implementation. As important as the number and extent of these changes is the time the design change is introduced. We hypothesize that the further system development has progressed, the more costly will be any change because of its broader implications and effect on completed work.

Although some measure of the cost effects could be secured by recording and gathering data on system changes, this highly significant factor is difficult to estimate in advance. Analysis of experience data might lead to identification of other factors that correlate with change, and are easier to predict.

(4) Extent of system dispersion and number of interfaces.

A dispersed system that has many communications requirements to be satisfied by a combination of the equipment and the computer program will be, in general, more costly to design than one that does not have so many interfaces. In the case of command and control, when the system involves many commands at several levels, it may be necessary to abstract, summarize, synthesize, or elaborate on information for the commands. Further, the number of information or command centers as well as their separation suggest the need for adaptation data* and the need for emergency or back-up configurations and procedures. Of particular significance to the operational design are the problems of compatibility introduced when more than one organization, e.g., military service or government agency, is to be a part of the system and when some centers use automatic data processing while others do not. Experience has shown that solution of compatibility problems, particularly if there is time phasing of capability at various centers, is time consuming and expensive, in terms of cost as well as design compromise. Also, the greater the number of personnel involved in performing different functions within the information-processing system, the greater the design cost because of the increased number of operator positions (which may use displays) and the increased number of inputs and outputs the system must handle.

Numerical measures for this factor are readily available. Simple counts can be made of number of centers, number of interfaces, and number of operator positions.

(5) Number of other components and subsystems being developed concurrently as part of the system, e.g., in a command and control system, sensor, and communication subsystems.

---

*Geographic and equipment characteristic data peculiar to one specific computer installation.

This factor measures the complexity and difficulty of developing a program system that must reflect both the design and operation of other subsystems that are, themselves, in a state of development. As such, the factor is closely related to expected number of changes (3) above. The parallel development of subsystems is often necessary to hasten the completion of the operational system. The greater the number of subsystems being developed in this fashion, the greater the cost. Obviously, this factor is directly measurable.

2. Program Design and Production

Program Design and Production includes cost factors associated with support programs, and the operational programs as determined by the constraints of personnel, hardware, and operational requirements.

In the program design activity, the operating system description and the operational specifications are used to create the detailed programming specifications. Design of the program system involves the determination of its broad logical subdivisions, the design of an executive program that controls the sequencing of subprograms, the design of the data base structure, the allocation of computer storage, and the specifications for any utility and support programs required.

The utility programs are the tools with which other programs are built. Some of them may be already designed and available, but usually some special tools will have to be designed and produced. Since preparation of utility programs may take many months, both this time and the associated cost must be considered in the costing of the operational program.

The factors in this category, as in Operational Design, center around the question, "How clearly understood are the (program) requirements?" Again, completeness, clarity, and detail combine to act as a damping factor on cost. Additionally, the program design factors include size and complexity.

(1) Number of computer program instructions and the types of programs that must be produced.

Sheer magnitude is a critical factor in allocating both resources: men and computer time. As a numerical entity, it is the basis for many of the rules of thumb currently in use by managers. Despite its importance, little research has been done to develop reliable methods for estimating the number of instructions. Very often,

these estimates turn out to be grossly inaccurate. One humorous observation on this situation was made by J. W. Garwick: "Programs do always get K times larger than a first thorough calculation indicates. I use K = π."*

In the design of business data-processing systems, the number of instructions has been poorly estimated. The Controller's Institute says: "From the detailed flow chart, the number of instructions required to carry out the operation must then be determined. This is possibly one of the phases which has caused the greatest trouble to most companies, since most EDP groups have at one time or another seriously underestimated the number of instructions required. Most companies report that their estimates become more accurate as they gain more experience in programming work, but they still are forced to do a great deal of educated guessing, based on a subjective evaluation of the complexity of the current operation as compared to one previously programmed....Every company we visited added a substantial safety factor varying from 20 per cent for a company which claimed, due to experience, a reasonable accuracy in its estimating procedures, to 400 per cent for a company which had found itself that far out on a previous estimate."**

An experienced estimator examining a proposed system that will have many similarities to previous systems may predict the number of instructions with some accuracy. This is the case with some command and control system programs that are designed in an evolutionary manner--that is, by a series of models or phases. For example, some cost guides for the 425L program for the NORAD Combat Operations Center have been developed based upon earlier versions of the program system. The instruction estimates are based upon number of registers for status and summary type messages, for display manipulation functions, for input/output tables, and so forth. Unfortunately, program histories and costs are not often documented in sufficient detail (such as this) to be useful in preparing such guidelines.

When no experience with a similar system is available, the number of instructions is much more difficult to predict, since it can be done, under the current state-of-the-art, only by weighing

---

*Leth-Espensen, J., On the General Problem of Compatibility of Computer Programs and the Particular Difficulties Embodied in Programming for Large-Scale Defense Systems, NATO, unclassified, ASTIA Document 273 710, p.51.

**Business Experience with Electronic Computers, New York, Controller's Institute Research Foundation, 1959, p.111.

subjectively many of the factors listed in this paper such as the performance of the compiler, the nature of the hardware, etc. One promising quantitative approach has been suggested by one of our associates.* For a limited sample of program systems, he has demonstrated certain consistencies in the relationship between the frequency of occurrence of the decision class of instructions and the total number of instructions in the program. This may be an important step toward estimating instructions directly from operational requirements.

With an estimate of the number of instructions, managers have estimated cost in terms of the number of man months and computer hours. (Number of instructions refers to machine language instructions.) In some prior work at SDC, it appeared from a small sample that the total number of man months is an exponential function of program size. Figure 2 is based upon the results of implementing eleven program systems. The dashed line is 200 instructions per man month, a frequently used rule of thumb for large programs. Attempts to explain this nonlinearity (the deviation from the rule of thumb) include recognition of such factors as increases in communication and coordination and increased complexity in the larger programs that may require an increased amount of labor.

In work on smaller programs or program systems (of, for example, less than 10,000 instructions), data showed rates ranging from 400 to 1000 instructions per man month for individual programs. A further analysis of the data shown in Figure 2 revealed that an average of 225 instructions per man month are produced for operational programs, and 311 per man month for utility programs (see figures 2a and 2b). One possible explanation for the lower cost of utility programs is that the program developer is the user and therefore can write his own requirements with little external coordination.

A similar investigation of the relationship between number of instructions and number of computer hours has resulted in the hypothesis that the number of computer hours used to develop a large-scale program is directly proportional to the program size. Figure 3 is based on experience with seven large program development efforts.** A statistical analysis of these data is given in

* Bleier, R. E., Frequency Analysis of Machine Instructions in Computer Program Systems, TM-1603, 19 November 1963.

**Management of Computer Programming for Command and Control Systems, Heinze, K., N. Claussen, and V. LaBolle. System Development Corp., TM-903, 8 May 1963.

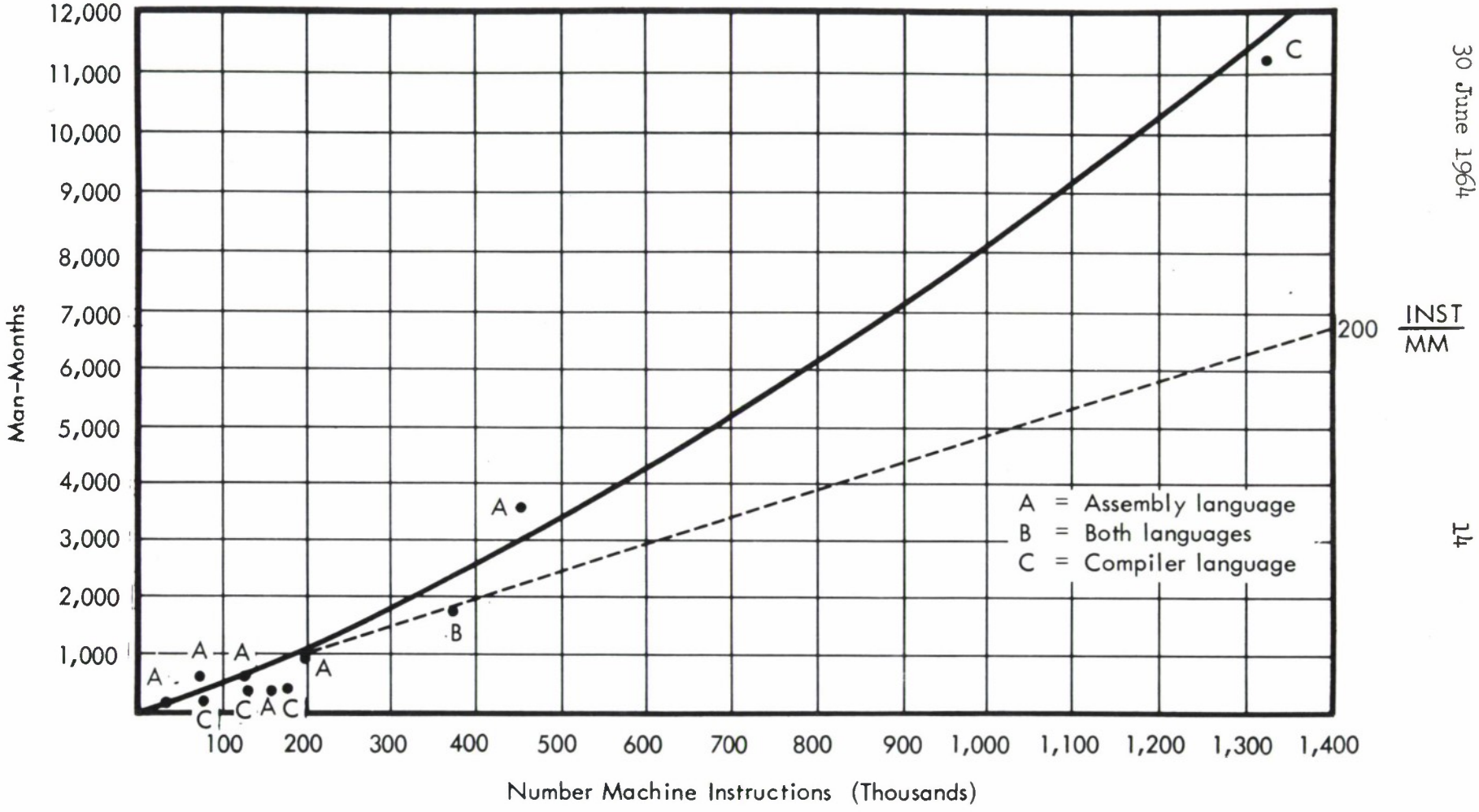


Figure 2. Man Months for Program Design, Production and Test Versus Program Size

This figure shows the costs of programming in man months as a function of the total number of instructions written (for both operational and utility programs). Because the data are so gross, no attempt was made to calculate an estimating function. The curve is an "eye-fit." For comparison, a dashed line representing a rate of 200 instructions per man month, a popular rule of thumb, has been inserted. In this range of program size, it appears that the cost of programming is an exponential function of program size. This may be due to such factors as increases in communication and coordination and increased complexity in the larger, interrelated programs.

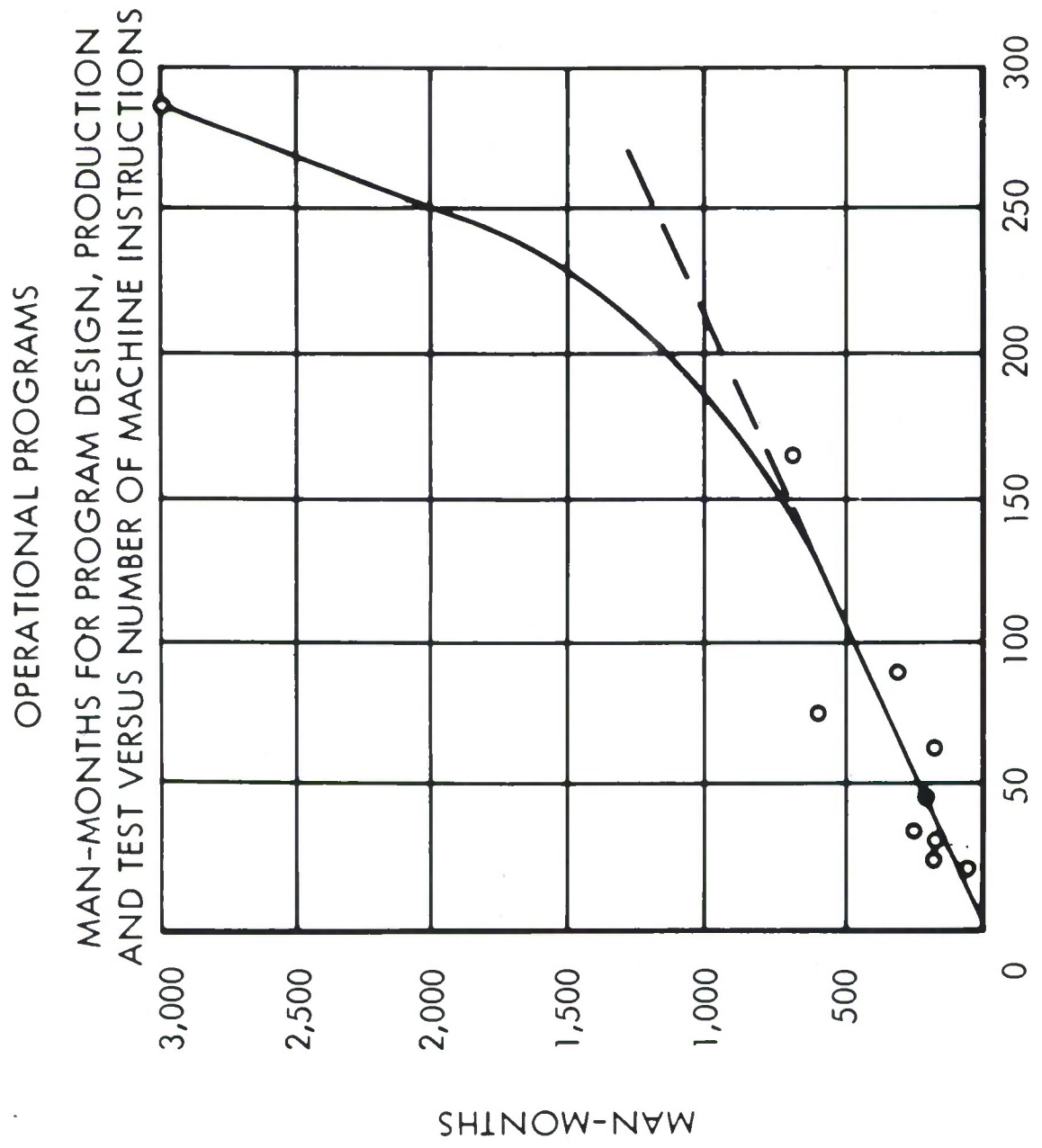


Figure 2a. Man Months Versus Number of Operational Program Instructions

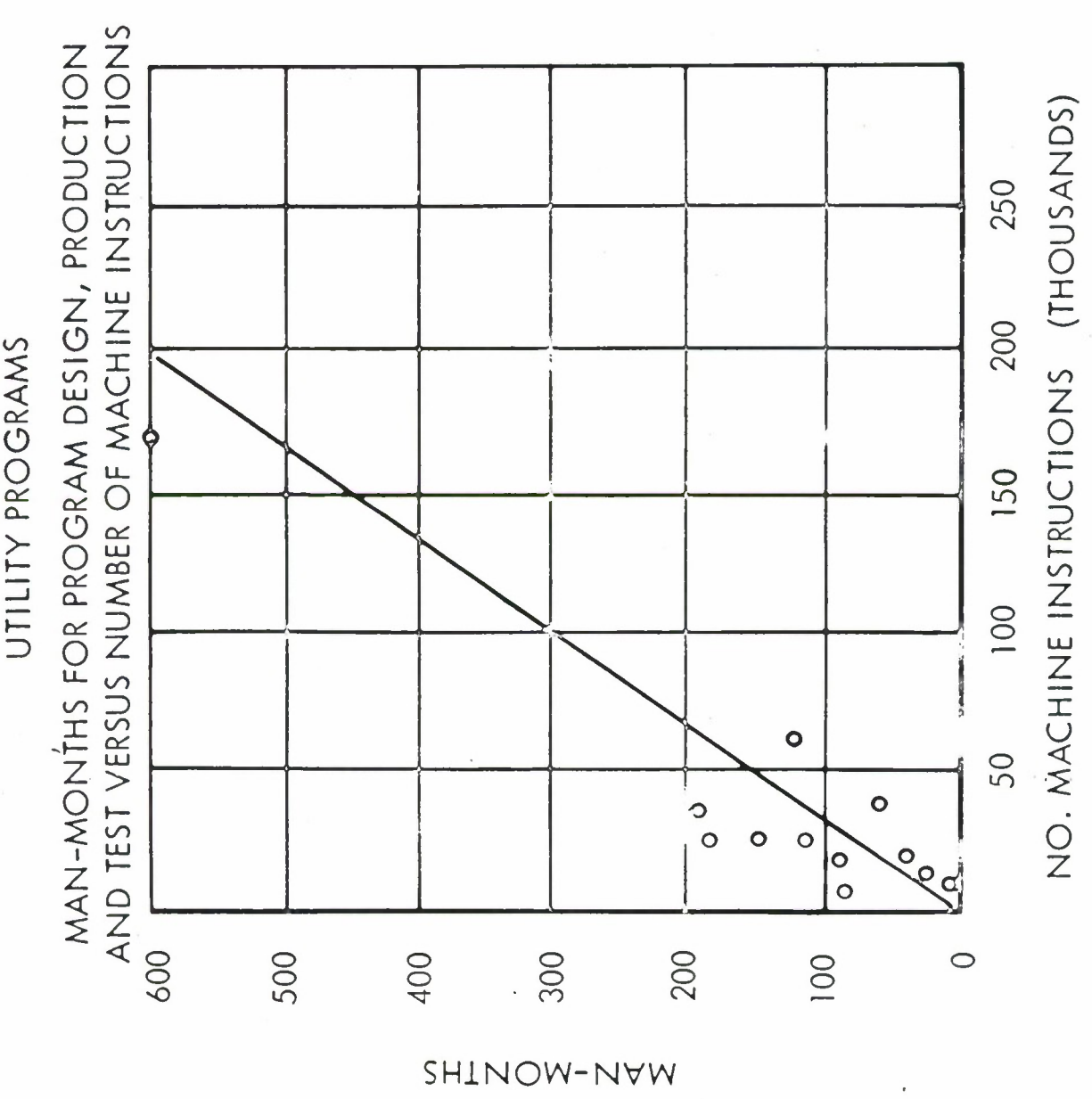


Figure 2b. Man Months Versus Number of Utility Program Instructions

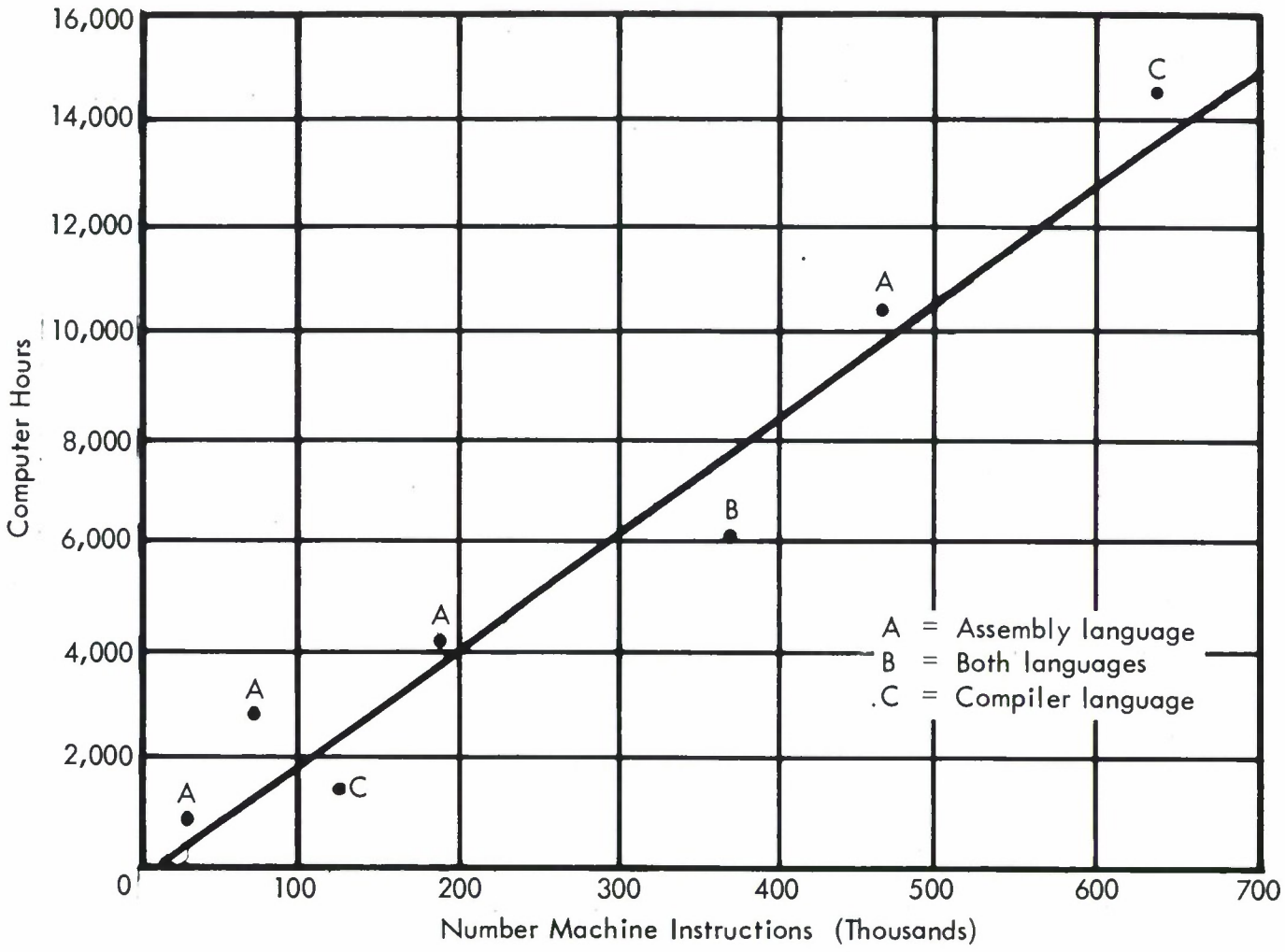


Figure 3. Computer Hours Used as a Function of Program Size

Appendix II shows the calculations to arrive at a least-squares fit to these data points. The resulting equation is $y = -174.3 + 21.7x$ where y = computer hours and x = machine instructions in thousands.

Appendix II to illustrate what could be done with a greater quantity of more reliable data. In this case, the resulting equation suffers from insufficient and unreliable data, and would be useful for estimating computer time only if an accurate means were available of estimating number of instructions.

In the case of the relationship of number of instructions to man months, the curves in Figures 2a and 2b are eye-fits and should be interpreted only as a first approximation, particularly since the data depicted were not entirely consistent. To be valid, these plots comparing data require clear definitions of program products, performance characteristics and quality measures. For example, in measures of number of machine language instructions, program systems should be classified as to the percentage of instructions derived from subroutines and from previous versions of the same program system.

(2) Number, types, and frequency of inputs and outputs to the computer(s).

In addition to influencing the cost of the program, the number and variety of inputs and outputs to and from the computer may be a clue to the size of the program needed. In any event, the greater the number of inputs and outputs, the greater the expense of both the operational and the program design. Increased input-output capability implies increased capability of both equipment and programs, and correspondingly, the increased analysis to insure compatibility with the input-output terminals.

Other input-output factors that influence cost are the rate of arrival (or departure) for messages and the amount of format conversion, i.e., pre- or postprocessing, needed. Higher rate of input-output is more costly, and obviously the more processing required for input or output data, the greater the cost.

(3) Extent of innovation required in the program system; that is, the degree to which programs are similar in nature to those previously written.

If programs to be developed will be similar to previously developed programs, such as in the mathematical, clerical, or logical areas, and this similarity is recognized, the cost of programming the new system will clearly be less. For example, the availability of reliable, well documented utility and support programs will help reduce costs. The primary point is that the development of new program applications will result in increased costs.

One approach to the development of new and unique programs is to test feasibility with experimental or prototype programs. Although it is costly to design and test a prototype or experimental program, this reduces the impact of innovation and so provides savings in over-all program design. It also clarifies requirements and operating procedures, and program development techniques, enabling the operational program to be developed with more confidence and fewer errors. The quantitative trade-off relationship between the cost of a prototype and the savings that result from it is not known.

(4) Number, types, and quality of publications and documentation for both customer and internal use.

Documentation is an inherent part of programming, though its cost and extent are often underestimated. The experience data shown in Figure 4 suggest that the number of pages of formal documentation is linearly proportional to the total number of instructions in the program.

Some rules of thumb used to estimate the amount and cost of documentation are as follows:

(a) Approximately 10.5 pages of documentation are needed per thousand lines of program code.

(b) A drafting rate is 3 to 5 pages (750 to 1250 words) per man day.

(c) Technical review rates of 20 pages per man day are average and may range to 50 or 100 pages per day.

(d) Typing rates average 15 to 20 pages per man day.

(e) Illustrations in line drawing (e.g., flow charts) average approximately 40 pages per man day, and revision of these, up to 80 pages per man day.

(f) Duplication of a large document is at a rate of about 25,000 pages per man day; this rate varies considerably with the process.

From these rules of thumb, a reliable prediction of documentation costs is feasible, given reliable knowledge of the over-all size and scope of the project in terms of documentation requirements.

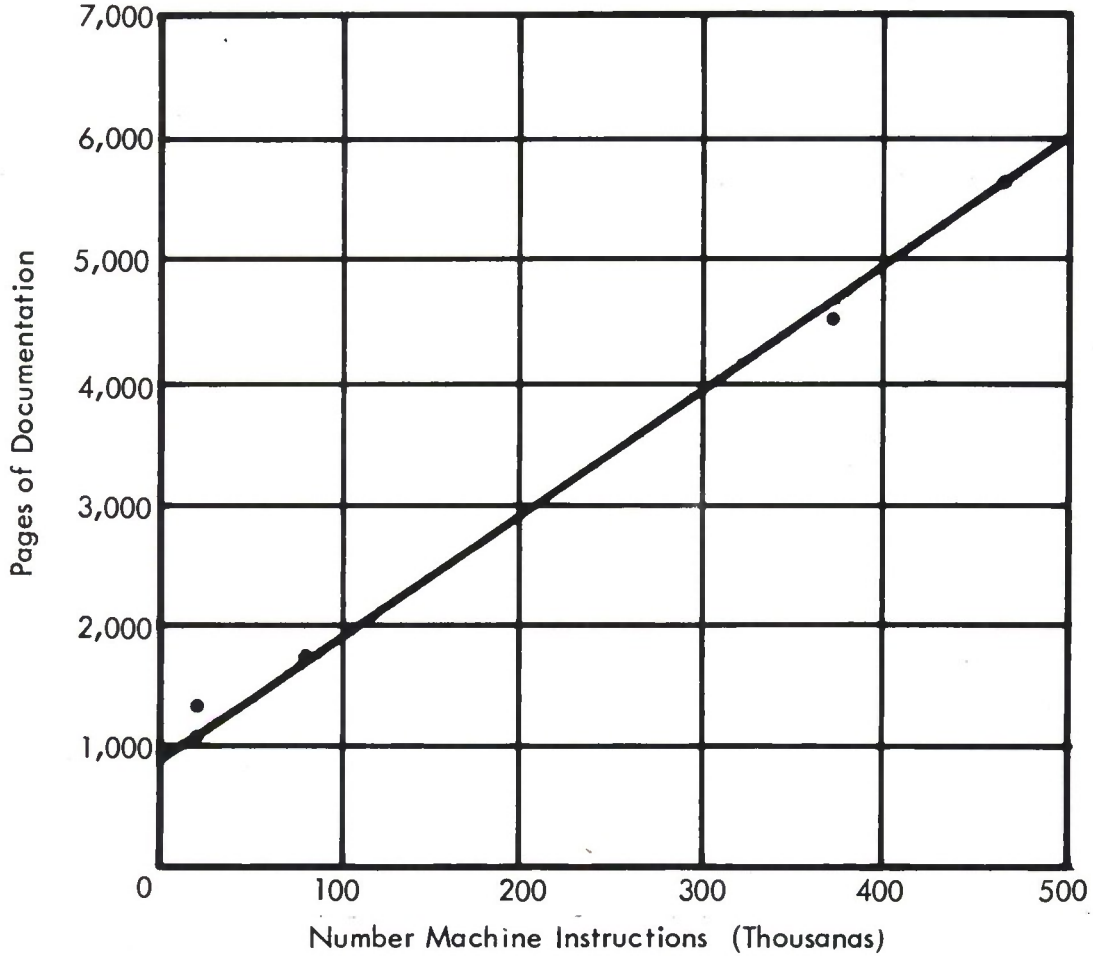


Figure 4. Number of Pages of Contract Required Documentation Versus Program Size

Data here are for five program systems. Pages of documentation refer to the number of contract pages of documentation actually produced. The graph suggests that program size in this range and number of pages of contract-required documentation are linearly proportional; it also illustrates the recognition of documentation as a major integral part of programming for command and control systems.

(5) Extent of complexity of the data-processing functions.

Complexity of functions is difficult to define. It appears to be associated with program design rather than coding, in that some programs have certain qualities that make them more difficult to analyze and design than others of equal length for the same computer. Several attempts have been made to define a so-called "complexity index."* Because most of the weightings in such indexes are somewhat subjective and because the elements of complexity themselves are hard to define, no generally agreed upon standards currently exist. Some of the elements of complexity might include the following:

(a) Degree of interdependence of subprograms.

(b) Number of decision points.

(c) Number of input-output requirements.

(d) Number and diversity of information-processing functions.

Another classification scheme has been suggested that assumes that the data to be processed could be scored by assigning measures to the following:

Class	Functions
Clerical	Store, retrieve, reformat
Synoptic	Reduce, classify
Predictive	Predict, forecast based on pre-established criteria
Directive	Decide, based on pre-established criteria and summing the product of class weight times the number of functions to be performed in each case.

A major problem with these schemes for assessing complexity is that they require that much of the program design be complete, or at least known, before they can be applied. In addition, since relation of statements of programming requirements to the complexity of programs remains to be done, the scheme sketched above may be of little value in helping to estimate costs early in the conceptualization process.

*For example, the work performed by Diebold Associates for NAVCOSSACT and the STEPS (STEPS: Staff Training Exercise for Programming Supervisors, R. Boguslaw, H. Richmond, and W. Pelton, System Development Corp., TM-321, 25 February, 1959) exercise both have described such indexes.

To isolate and define the costs of complexity, an experiment could be designed that would subject programmers with similar backgrounds (experience and skill) to a variety of programming requirements that had been ranked by panels of "experts" in their degree of complexity. Measurements could be taken on the time required by both men and computer to design and develop the programs. (A shortcoming of experimental approaches to answer many of these questions is that the cost of such tests might be prohibitive.)

(6) Degree to which the following program design characteristics are recognized and must be incorporated:

(a) Maintainability--the ease with which program errors can be detected and corrected.

(b) Changeability--the ease with which new functions can be incorporated into the program.

(c) Usability--the ease with which personnel other than designers can use the program.

(d) Flexibility--the ease with which the program can be used for other purposes with only slight modification.

These somewhat arbitrary characteristics of computer programs are often implied in the program requirements, but not explicitly defined. When programmers consciously attempt to incorporate these characteristics into the program, there may be an increase in the cost of the design, but the resulting program, in the long run, may be less expensive to operate and maintain. This factor is characteristic of the difficulty both users and programmers have in describing the performance characteristics of a computer program. An analysis of existing systems might help define these characteristics and contribute to the adoption of standards to aid program design and production.

(7) Extent of the constraints on program design.

The more constraints upon the program design, the more costly it will be. There are a number of ways that program design is constrained, requiring a more sophisticated or clever approach, e.g.:

(a) Computer storage capacity.

(b) Number of input-output channels.

(c) Timing of internal transfers.

(d) Extent to which the program must operate in a real-time mode.

The exact effect upon costs of various types of constraints is currently unknown; experimentation might help determine such effects. For example, it would be possible to determine how much longer it would take to do a given program under the constraint of, say, 1000 primary memory locations versus one of 2000 locations.

(8) Number, size, frequency, and timing of program design changes.

Program design changes are those that occur after the completion of the program design phase. These changes result from changes in either the operational functions themselves or the methods of performing them and almost always increase programming costs. Some changes are simple and localized, while others have a "ripple," or "snowballing" effect,requiring, in addition to the change in one program area, changes to interfacing and interrelated programs. Every change requires examination to determine whether it has this "ripple" effect. Costing a change is similar to costing a new program, except that more details are available and the problem is to determine how much has to be redone; e.g., retesting and document revisions are almost always necessary. Unfortunately, very little data exist on the cost of changes.

(9) Extent to which data for the data base are available, or data collection is required.

The availability of data for the data base or, at least, the availability of the format of the data (e.g., categories, ranges, maximum and minimum values) will allow data base design to proceed early in the design phase, and thereby minimize the expense of later redesign if the data are not immediately available. Ideally most of the data should be available before program design begins, although data collection continues throughout the implementation process because of changing system requirements and new component specifications. If the data are not provided by the customer, and the programming contractor assumes the responsibility for data collection, a substantial cost must be anticipated. Similarly, classified data require special handling procedures that increase the cost. Because of the rapid rate of change of system data, the user must identify the appropriate sources of information, and both the programming contractor and user must provide and participate in a procedure for certifying the validity of the data and for concurring on formats and data elements. An estimate of the percentage of data that is available can usually be made early during the planning.

(10) Number of entries (total size) for the data base, the number of different types of data needed for it, and the extent to which each can serve many programs or subprograms.

The greater the total amount of data and number of types of data to be handled, the greater the cost. Once the content and structure of the data base are known, an estimate of storage requirements can be made. This should include recognition of possible ways to store data, e.g., a determination of the number of central tables containing common data used by many subprograms, and isolable tables containing data used by only one program. The central tables, of course, are most desirable in terms of reducing storage requirements, although their use may require additional programming. The number and variety of entries in the data base may give some indication of the complexity of the data base design and the extent of the data dictionary required in addition to the direct cost relationship stated above.

(11) Efficiency of the programming language and the compiler or assembler.

Although the relative merits of various procedure-oriented or symbolic assembly languages have been discussed at length, they have been subjected to relatively little systematic, quantitative research. The hypothesis that procedure-oriented languages (POL) are easier to learn and easier to use, and allow greater productivity (i.e., instructions per man month) in the coding and test phase has not been proven conclusively, but can be borne out in some specific cases. For example, the survey in TM-903* indicates that an average productivity rate for command and control programs using symbolic programming languages is slightly less than 200 machine language instructions per man month, while data collected on certain programs written in JOVIAL and NELIAC reveal productivity rates of about 450 machine language instructions per man month. Possibly offsetting this gain is the suggestion that the number of machine language instructions generated by compilers to perform a given set of logical functions will be greater than if they were prepared by an assembler. Other suggested advantages of POL are reduced costs for reprogramming new machines and increased management understanding. We have no data on hand to help confirm or deny these advantages, but they must be traded off against the possible cost of developing and maintaining these languages and their compilers if it is not possible to use one in existence. Although previously high, the development costs of compilers seem

*Heinze, et. al, op. cit.

to be decreasing, apparently by use of improved program design techniques and "bootstrapping," i.e., using a compiler and POL to develop a new compiler, usually for a different machine.

In addition to the productivity rate measured in instructions per man month, other measures of the efficiency of the programming language and compiler are the computer time necessary to compile the source program and the computer time required to execute the object program. A comparison of these parameters with the symbolic assembly process reveals that the increased productivity of the POL has been traded off against the cost of increased computer time. Current compiler developments have had, among other goals, that of reducing compilation time. New designs indicate success in meeting this goal. Also, increased time for compilation may be more than offset by decreased computer time for debugging.

The Air Force Assistant for Data Automation (AFADA) conducted some tests, primarily to provide a basis for adopting a standardized procedure-oriented language rather than to assess the advantages of POL's in comparison with symbolic assembly languages. These tests were designed to compare the performance of nine POL's. Nine programmers, each using a different POL, programmed the same problem and their programs were compared to one written in a symbolic assembly language. While criticism was leveled at statistical significance and experimental design of this test, because no attempt was made to measure programmer variability, the results indicated that a good machine-oriented language can produce a more efficient program (i.e., with fewer machine language instructions and shorter execute or operating time) at greater programmer expense than the average POL.* More study and/or tests are needed to develop hard data on cost effectiveness of programming languages.

The question of efficiency of the language is more often implied in discussions of the compiler, but can be examined explicitly. For example, ease of use may determine the number of errors produced and the extent to which debugging can be done in source language to help to reduce the cost of using the language.

*Other work on language standardization is documented in RAND Memorandum RM-3447-PR, Programming Languages and Standardization in Command and Control, J. P. Haverty and R. L. Patrick, January 1963, and TM-688/000/01, Computer Programming Standards in Command and Control, 15 February 1962.

(12) Extent to which programming tools are available and usable.

In general, the availability of reliable programming tools for the program development staff tends to reduce program development cost. Some examples of useful aids are:

(a) Data tools, such as data description languages, table generators, table design programs and format and list description tables.

(b) Program modification tools, such as design change, parameter change and error-correction routines.

(c) Control tools, such as accounting or bookkeeping routines, interrupt and restart programs and error-detection techniques.

(d) Special tools, such as hardware diagnostic routines, data base manipulation procedures, and loading and editing techniques.

To get a rough measure of this factor, a complete list of tools needed could be used to find percentage of tools available.

As mentioned earlier, if there is a need to develop these tools, this cost must be traded off against their usefulness as measured in man months of labor and machine time saving.

(13) Extent of the completeness and clarity of the system test and acceptance test requirements.

The objective of system test is to identify and eliminate all component interface problems and to verify that the total system, operating in a live environment, performs in accordance with the operational specifications. Beginning after the completion of operational design, development of a comprehensive system test design is a costly and complex process and requires participation of the customer and all component developers. While actual conduct of the system test may represent a small effort for the programming contractor, the analysis of test results is the most difficult and costly part of the process. For example, component failures are often detected, but insufficient information exists to identify and correct them. Furthermore, the components may be the responsibility of separate subcontractors, which presents a difficult and costly coordination problem. Extensive requirements for acceptance tests and demonstrations result in increased costs because of extensive preparations, planning and coordination.

B. THE RESOURCES WITH WHICH TO DO THE JOB

This section addresses the influence of the principal resources, data-processing equipment and personnel, upon cost.

1. Data-Processing Equipment

The Data-Processing Equipment category includes cost factors associated with the hardware required to produce and test a program, including all input, output and peripheral equipment.

The lack of techniques* for analyzing computers and the rapid, continuous development of new computer hardware preclude making statements about the effect of hardware upon programming that will be valid for more than a few years. For example, a recent comparison between a typical 1954 electronic processing system and a typical 1962 system showed that internal memory capacities have increased over 100 times, add times have decreased by a factor of 30, input-output speeds have increased by a factor of approximately 11 and so on. Current equipment systems use components that were not available several years ago, e.g., cathode ray display tubes and disc memories. Thus, any analysis to identify the long-range effects of hardware upon programming cost should include time as a variable. However, the identification of cost factors below accounts for some of the variation in equipment capability and also considers factors that appear to be meaningful regardless of changes in equipment design.

(1) Number of hours per day of computer availability.

A commonly held notion is that the more hours per day the computer is available to programmers, the lower the over-all cost of the programming effort. Certainly, it is to be expected that greater computer availability will cut down the total number of man months of programming time required. And if computer time is very inexpensive compared with programming time, it obviously pays to strive for the most hours of availability.

Among the considerations in determining the number of hours per day of computer availability for a given effort are the following:

(a) Number of shifts per day of computer operation.

(b) Time required for preventive maintenance.

---

*One such technique was reported on at the 1964 Spring Joint Computer Conference: "The Use of a Computer to Evaluate Computers," D. J. Herman and F. C. Ihrer.

(c) Unscheduled downtime (reliability).

(d) Competition, or number of other computer users sharing the machine.

A new development that promises to reduce some of the competition for programmer use of the machine is time-sharing, a technique that enables several people to work on the computer at a given time, each with the feeling that he has complete command of the system. The effect of time-sharing on programming efficiency and costs has yet to be analyzed.

Delays in the delivery of the hardware may lead to additional costs. For example, one computer arrived four months after the scheduled date, requiring negotiations for use of, and programmer travel to, substitute machines, for testing. In another case, programmers had difficulty getting machine time on the designated computer and had to do most of their testing on an alternate machine. Whenever this occurs, particularly with alternate computers of slightly different configurations, the over-all programming effort will take longer and be more costly.

(2) Extent of capability of the computer and its suitability for the job to be done.

Although greater capability costs more, from the over-all system point of view, it may considerably decrease the programming time required. For example, larger memory capacities make programming easier. Among the capabilities to be considered here are the following:

(a) The power of the order code.

(b) Capacity and access time for primary and secondary memory.

(c) Operate time.

(d) The speed and availability of input-output equipment.

(e) The number of index registers.

(f) Multiple- versus single-address.

Few data exists to aid in estimating the effects of these factors. Although it has not been done to date, data could be collected concerning the relative cost of programming the same jobs using a variety of machines and configurations. One would expect that the computer manufacturers would be vitally interested in collecting such information.

(3) Extent to which the operation of the computer and peripheral equipment is reliable, well tested and well documented.

Excessive computer downtime lengthens the total programming man-months by more than the downtime itself because of the disruption of schedules and plans. This requires effective, timely, on-site emergency and preventive maintenance.

If the operation of either the central computer or peripheral gear is not well documented, it is difficult for a programmer to know whether errors in his program result from machine characteristics or from his own mistakes in logic. Sometimes delays are encountered in contacting the equipment manufacturer to learn about equipment characteristics that should be available in documentation. Documentation that is inaccurate or misleading is as bad or perhaps worse than no documentation.

(4) Number of automatic data-processing components being developed concurrently with the program.

Automatic data-processing components include the computer and all those pieces of equipment that can be recognized, addressed or controlled by the computer program. If any of them are not available at the time the programming starts, but rather are being developed concurrently, it is necessary for the programmers to write their programs based upon equipment specifications that may change as the design of the components changes throughout their development process. Additional costs are incurred in maintaining this communication channel with the equipment developers. Unless these changes are communicated to the programmers accurately and quickly, the final programs will not operate satisfactorily, and the program implementation will be delayed.

(5) Number of different computers for which programs are being prepared.

If the system includes several computers from different manufacturers or with slightly different configurations or requirements for communication between them, the programming effort will be more costly, because programmers must become familiar with different hardware characteristics and additional work must be done to ensure compatible operations.

(6) Number and types of displays used.

It is probably safe to say that the addition of a variety of display devices (e.g., wall display, individual display), particularly those requiring complex input descriptions, will increase programming costs. There is information on the extra effort required for programming various types of displays, but these data have not been collected or analyzed.

(7) Extent to which adequate EAM support will be available.

If adequate keypunching and other EAM support is not available, and must be subcontracted to outside agencies, this adds to the cost of programming, and introduces time delays in the testing schedule.

2. Programming Personnel

This category includes cost factors resulting from the direct labor needed to develop a computer program.

One measure of cost proposed in this paper is man-months. Clearly, the quality of the personnel and the working relationships among them will influence the number of man-months required to perform a task of any given size. The factors below reflect consideration of this influence:

(1) The types and quality of programmers.

This is certainly one of the largest contributors to the over-all cost of a computer program. The nature of the job determines the appropriate types of personnel and it is likely that the proper mix will vary somewhat from job to job. In most organizations, programmers are ranked in several classes by such titles as coder, junior programmer, senior programmer, system analyst, and so forth, reflecting various levels of skill, experience, and ability to assume responsibility.

There are three particularly important types of programming experience:

(a) Experience with the particular computer--The more experience a man has with the particular machine for which the program must be developed, the more familiar he is with its capabilities and the less time he requires to program and test.

(b) Experience with the particular language--Programming languages differ in their suitability for various programming efforts. Experience with a procedure-oriented language, such as JOVIAL for command and control, or COBOL for business systems, eases the job for the programmer and thus requires a smaller effort (i.e., number of man months).

(c) Experience with the particular application--If the programmer has experience with the particular type of system being programmed, and the particular problems of the user, fewer man months will be required for the new effort.

Although the assessment or measurement of quality or skill requires more than consideration of the number of man years of experience, reliable methods that permit effective comparison of programming personnel have not been developed. Several efforts to attack this problem are underway. For example: At the University of Southern California, a study sponsored by the Navy is seeking to develop optimal personnel selection and classification procedures by analyzing the job of the computer programmer and developing criterion measures of performance.*

(2) Number of man months of programmer training required.

The high demand for programmers makes training necessary as a support activity for program development. The effect of programmer training upon cost may take the form of a U-shaped curve. That is, for any given task, there is an optimum amount of training that the programmers should have in the particular language and the application. More training than the optimum merely adds man months to the project without producing a commensurate return, while insufficient time in training presumably leads to errors and confusion in the programming.

There are many components of the cost of training. First, of course, is the actual time spent by the trainee in class instead of doing productive work which depends partly upon the experience and quality of the trainees and partly on the nature of the programming work to be done. Secondly, the time spent by the instructor in preparing for the class may be very costly, particularly if the information to be taught is inadequate, poorly

*Rigney, J. W., R. M. Berger, and A. Gershon, Computer Personnel Selection and Criterion Development: I. The Research Plans, Los Angeles Department of Psychology of the University of Southern California, February 1963.

written or disorganized, as may be the case in courses for computers that are still in the development or prototype stage.

Data that exist on training costs in the personnel records of programming organizations have not yet been collected or analyzed. To study the cost-effectiveness trade-off, it would be possible to conduct an experiment in which programmers with similar backgrounds would be divided into a number of control groups each of which would receive a different amount of training after which each group would produce the same program(s). The cost of training could then be compared with the cost (i.e., man months, computer hours) saved as a result of the training.

(3) Number of programmers to be assigned to a given function or task.

A study to determine the optimum work group size for each type of programming effort has not been conducted. Some guidance is available from the Controller's Institute, however, which reports that "the general trend seems definitely toward the smaller units, individuals, two- or three-man teams, or an arrangement where two programmers work individually and then review and check each other's work."* Outside the programming field, Ellis A. Johnson has stated that the minimum time of accomplishment in research and development occurs at a work group of between four to seven people.**

One practice that appears to have some success is the assignment of a clerk or trainee to assist the more experienced programmers in performing some of the tedious work. Also, the deliberate assignment of test responsibility to another individual or group appears to have had some success.

Experiments could be designed and conducted to determine optimum work group sizes; however, the experiments might be difficult to control, and costly.

---

*Business Experience with Electronic Computers, New York Controller's Institute Research Foundation, 1959, p. 111.

**Johnson, Ellis A., "A Proposal for Strengthening U. S. Technology," in Operations Research in Research and Development, edited by Burton V. Dean, New York, John Wiley & Sons, 1963, p. 34.

(4) Policy of obtaining and phasing of personnel to staff a new program development.

There are essentially two ways that personnel can be acquired for a new programming effort; they can be hired or they can be transferred from other jobs. From an over-all organizational viewpoint, the transfer of personnel from another job may involve a hiring cost to replace them. Depending upon the skill classification involved, hiring costs may be several hundred dollars or several thousand dollars per person.

In addition to the cost of obtaining personnel, there may be additional cost from poor scheduling or phasing of the project. Usually there is a gradual build-up, a peak and a phasing out of personnel in each of the activities of program implementation. With poor scheduling, there may be idle hands when it is too early to use them effectively or there may be an inadequate number of personnel at a critical juncture.

(5) Rate of turnover.

This factor is the percent of the work force terminating and being replaced per unit time. It affects cost in that the terminations or resignations must be replaced by either new-hires or transfers--usually people who are less experienced in the given task than the ones who left. Data concerning the percent turnover are usually available from personnel departments.

Again, an experiment to determine the effect of changes in personnel during a programming effort could be designed. Such experiments have been conducted for other types of task groups, but this type of research is still on the frontier of psychological and organizational research.

C. THE ENVIRONMENT IN WHICH THE WORK IS DONE

Factors that arise as a result of the available facilities and conditions under which the work is to be done are included in the following categories concerned with procedures, environment, facilities, services, indirect labor and overhead factors.

1. Management Procedures

The Management Procedures category includes cost factors associated with the plans, policies, practices and review techniques used in the administration of all phases of program development.

A general comment can be made regarding the majority of factors in this category. The cost of designing and instituting clear-cut procedures for the use of the computer or the submission of progress reports often seems high at the outset; but its true cost and value must be determined by comparing the cost of the plan with the cost of not having a plan or procedure. Unfortunately, it is extremely difficult to determine the cost of having a plan, and probably impossible to determine the cost of not having one. Nevertheless, experience on past contracts indicates that well planned projects enjoy higher productivity rates.

(1) Extent of use, maintenance, and monitoring of effective management plans within both the customer's and program developer's organizations.

This factor simply emphasizes the fundamental management principle of documenting plans and procedures to decrease costs by eliminating uncertainties concerning responsibilities and the source of decisions. Among the procedures and plans needed in program development are the following:

(a) Communication with other agencies.

(b) Concurrence on design specifications.

(c) Cost control.

(d) Management control in the form of PERT or Gantt charts.

(e) Document control (e.g., design file).

(f) Standards for coding, flow charts, etc.

Similarly, the customer must have a well defined management concept or plan for developing the system in which the programs will be embedded. This plan must include a clear statement of job responsibilities for all agencies involved, and a well defined channel of communication for all organizations involved. Some cost reduction stems from designating official representatives as points of contact to ensure the correct and rapid transmittal of information.

Perhaps as important as the plans is the reporting system that ensures that the plans are followed. Although not known in a numerical sense, it is a logical hypothesis that the cost effectiveness or value of an internal management reporting system follows a U-shaped curve (i.e., an optimum number of internal reports will yield a decreased total cost). Figure 5 displays this hypothesis of cost and value for internal reports. On the other hand, the cost of external reports, which are usually for the customer, simply increases the total cost linearly with the number of reports with little or no value to the contractor.

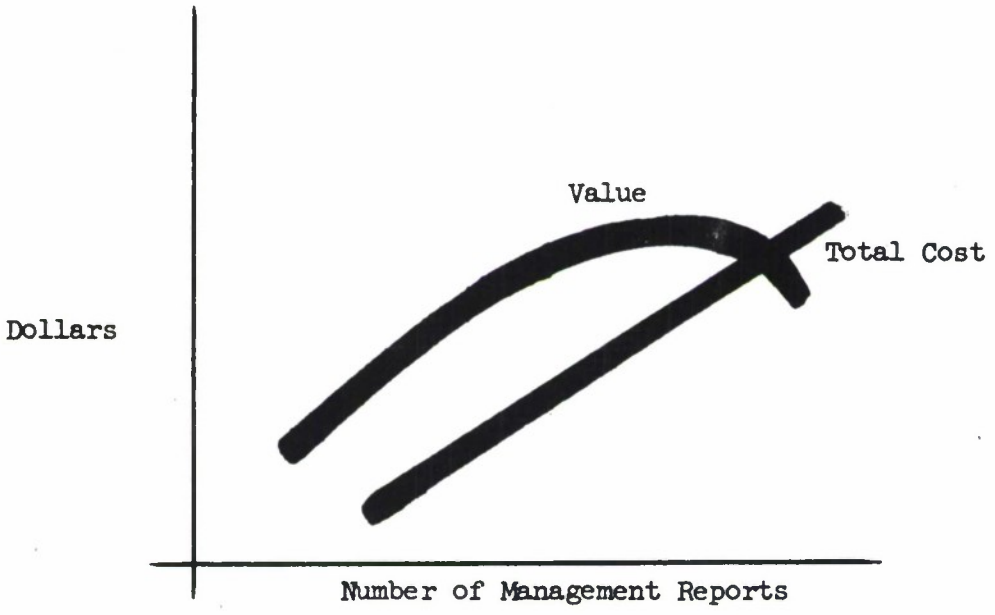


Figure 5. Hypothetical Relationship Between the Total Cost of Management Reports and the Resultant Value to the Programming Contractor

(2) Extent of formalized procedures for use of the computer facility.

An example of this factor is the policy set by management concerning the distribution of computer time to each programmer. A number of diverse theories exist concerning the effect of this factor on programming cost. For example:

(a) Maximum desk-checking and a minimum number of computer runs will minimize cost.

(b) A trade-off exists between desk-checking and computer testing such that an optimum number of computer runs per day will minimize total cost. Some data suggest that the optimum number may be two or three.

(c) An unlimited number of computer runs per day will produce a program in the shortest elapsed time.

An attempt to use this third theory and simultaneously realize maximum use of the computer facility is one basis for the concept of computer time-sharing. There seems to be little evidence or data to strongly support any one of these theories, and this factor is ripe for experimentation and further study.

A second example of computer usage theories are the two philosophies of program testing:

(a) Parameter test, or debugging of the subprograms, should be as complete and thorough as possible before assembly test, where several subprograms will be tested as a unit.

(b) Parameter test should only be done in a gross fashion, and assembly test should be entered into as quickly as possible to locate more errors in a shorter period of time.

A third example of formalized procedures relates to the consideration of open- or closed-shop operation. While closed-shop operation may minimize computer time for a given job, open-shop operation may minimize programmer time. Therefore, some trade-off exists between the two, and the formalized procedures to use the computer should take this into account.

(3) Extent to which there is a well defined and controlled system change procedure.

The fact that systems are developed in an evolutionary way is readily accepted, but effective techniques for accommodating changes during program development are not well known. If provisions for change haven't been included in the original design, even a small change, such as the addition of a few sensors in a command and control system, may necessitate extensive re-programming, retesting, and rewriting of documents. During development, frequent design changes lead to heavily patched programs that may not run efficiently. In the extreme, such programs may eventually have to be rewritten entirely in the form of a new model. Regardless of their impact on the program system, changes demand continual appraisal and hence, a mechanism and some effort devoted to these examinations.

(4) Extent of an error-reporting and -correcting procedure.

Because of the high degree of interdependency among subprograms in a large program system, an error in one subprogram may easily affect many others. An effective error-reporting procedure ensures that all programmers who may be affected by changes and program corrections are notified. In the absence of such a procedure, program changes may be made late in the development cycle when they could have been made earlier at lower cost.

(5) Extent of contingency plans in the event that the computer is overloaded or otherwise unavailable.

Often the computer may be unavailable for any one of a number of reasons (e.g., down for unscheduled maintenance, other user's priority, failure to deliver on schedule, etc.). In this event, a plan that designates the availability of an alternate computer (e.g., that of a service bureau or of another contractor) will save time that would be lost in waiting for the computer to become available or in searching for an alternate computer.

(6) Extent of quality control that is exercised during testing (e.g., reliability requirements).

Once more, no clear relationship exists between the cost of instituting quality control procedures and the penalties for having none. Poor-quality programs (e.g., of high error content) are more costly to install and maintain. The hypothesis of the U-shaped curve may also apply to the extent of the quality control procedures implemented in the programming process (see Figure 5). The question of definition of quality has been discussed under program design factors.

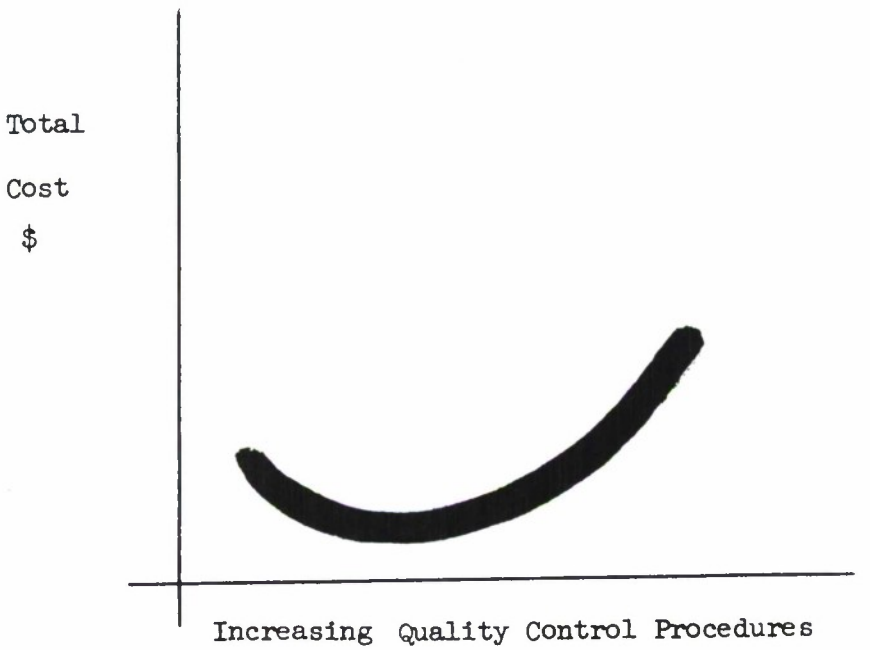


Figure 6. Hypothetical Relationship Between Total Cost of Program Implementation and Increasing Quality Control Procedures

2. Development Environment

The Development Environment category includes cost factors resulting from relationships with external organizations, including customers and contractors.

In general, problems in the relationship between a contractor for program development and the customer arise from insufficient understanding of the magnitude and scope of information-processing system development, the role of data processing, and the program developer's need for detailed requirements. As a result, the interdependencies that require coordination, concurrence, and data transmittal may lead to situations that increase cost of the program development and/or reduce quality of the products. The following factors are both symptoms and causes of such problems, and as such are somewhat overlapping and interdependent. It is significant that the majority of these factors will not be known at the contract proposal stage, but will be learned only after a contract is awarded.

(1) Number of agencies with which the programming contractor must deal and their level of experience with system development.

Coordination and concurrence seem to multiply as a function of the number of agencies with which the programming contractor must deal. Such agencies might be the (a) user or customer (e.g., NORAD, ADC), (b) contracting agency (e.g., MITRE, ESD), (c) system manager (e.g., Aerospace, MITRE), and (d) other subsystem-developing agencies. Deliberate plans for coordination may reduce its costs. Also, the more experienced the various agencies are in information processing system development, the less costly will be the coordination, with respect to program development.

When the program developer and the user are unfamiliar with one another's procedures, they should plan to educate one another. The cost of briefings and meetings to educate each organization should be considered an investment early in the implementation process to eliminate costly problems later.

The extent of system development experience of the user personnel, e.g., military, is important. Although it is difficult to evaluate its cost-contributing effect, this factor implies that increased costs stem from assignment of inexperienced user personnel to develop computer-based systems. In many cases, the assignment may be made deliberately so that personnel can gain experience.

Finally, a factor that is both important and readily measurable is the number of other agencies involved directly in program development. Several organizations (contractors) may participate, and increased costs will stem from satisfying the need for increased coordination and communication.

(2) Average number of days and effort required for concurrence.

Achievement of understanding of the projected operational design of the system and the formal agreement to it by the user is known as concurrence, which triggers the major effort to realize the program system. Indecisiveness in concurrence on design plans and specifications, or ignorance of responsibility for concurrence by the user leads to uncertainty, loss of time, and perhaps repetition of costly program design work that was underway during the period of delay. The program developer may undertake various tasks to develop the understanding needed and the schedule will contain an interval for review of specifications, but many times inadequate attention is given to the review and scheduling.

(3) Travel requirements.

Significant expenditure in a system development project may be for trips required for briefings and conferences for these purposes:

(a) Information and data gathering.

(b) Training and familiarization.

(c) Concurrence on requirements and design.

(d) Problem solving.

Also, travel or relocation may be necessary because the programmers are required to work at specific sites either because of customer requirements or computer availability. A remotely located computer facility will also increase costs because of communication problems and inconvenient access. Although this problem can be somewhat ameliorated by the use of remote input devices such as data link, data phone, or teletype, relocations may often be necessary.

There is a strong tendency to underestimate travel or to reduce it for economical or political reasons. If adequate funds are not available for travel, delays in getting information or concurrence may make the job quite costly in terms of work that is performed incorrectly and must be corrected later. The total amount of travel is a function of the number of sources of information,

number of associated developmental agencies, geographical location of customer, geographical decentralization of operations, computer location, and other factors.

Also, an obvious but sometimes overlooked fact is that when personnel are travelling, they are not available for activities such as program design or test. Furthermore, such trips often require preparation and subsequent trip reports, so that more time is taken away from the activities that nominally constitute program development.

(4) Extent to which delivery dates for required programming tools are reliable, and correspondingly, the amount of pressure caused by a tight schedule.

These two factors reflect the concreteness of the schedule and the pressure it may impose upon the programming personnel. If a schedule of delivery dates for required tools is not reliable, the developer can expect to experience costly delays. He can expect his schedule to slip on a day-for-day basis with the slippage in the delivery of the required tools. This factor, combined with the possibility of an already crowded schedule, will cause total costs to increase nonlinearly if overtime is necessary to maintain the original schedule.

(5) Extent to which the computer is operated by another agency.

With respect to a development of a particular program, ideally, the developer should operate and control the computer facility. If not, increased costs may result from insufficient computer time, undesirable distribution of scheduled hours, and delays due to inconveniently scheduled equipment modification.

3. Facilities, Services, and Supplies

The Facilities, Services and Supplies category includes cost factors related to supplies, physical plant, indirect labor and overhead.

In most accounting systems, normal overhead and miscellaneous supplies are covered by a percentage addition to the estimated direct labor and materials. Such an overhead figure includes normal costs of office space, stationery, pencils, top administrative management, personnel services, plant maintenance, and so forth. However, in a large-scale programming effort, a number of unusual expenditures might fall into this category and have an

effect on cost in excess of that represented by the expenditure itself or by the allowances made in an average overhead/burden rate.

(1) Number of computer operators and EAM personnel required.

In using man months as a cost measure, we refer specifically to programmer man months. Potentially substantial, the cost of support personnel may be considered overhead and may include the cost of computer operators, keypunch operators, operators of off-line equipment and others associated with the computer. Such personnel provide services for the programmers and free them for more direct work, and, in addition, contribute to more effective use of the computer. No data exist concerning the optimum ratio of such personnel to programmers although it is conceivable that such a ratio could be determined from an analysis of support activities and the cost of having programmers perform such tasks.

(2) Number and experience of technical management personnel, administrative personnel and technical editors.

Programmers need various types of support to function effectively. Good management is clearly one of the most important factors.

If available, management personnel with appropriate experience in information-processing-system development will help reduce cost. Only intuitive notions exist about the degree to which poor management adds to cost. For example, poor management may lead to delays due to poor planning, inadequate coordination of programming efforts and customer requirements, delays in concurrence, or decreased quality of products.

Administrative personnel include typists, secretaries, executive assistants, and so on. Each organization has its own history of the most efficient mix of technical and administrative personnel. For cost computation purposes, figures for the ratio of technical to administrative personnel are usually available from accounting departments. Here again, we would expect a U-shaped curve, i.e., there is some optimum number of administrative personnel above which the expenditure for their talents is too costly, and below which more expensive technical personnel may be required to do nontechnical chores.

Similarly, each organization has its own experience in the number of technical editors required for reviewing program documentation. Because of the importance of documentation, one current trend is to identify technical writing as one of the skills or talents of the programmer. No data exist concerning the degree to which assistance from a technical editor frees the programmer for other tasks or prevents ambiguities and errors from appearing in documentation.

(3) Cost of special simulation facilities, computer room facilities or special office equipment.

Installation of a new computer usually requires expensive site preparation, e.g., special wiring and air conditioning, false flooring, space for storage and movement of parts and equipment, maintenance and test hardware. To estimate this cost, one authority offers the following rule of thumb: "In practice, a figure of $100,000 to $150,000 seems sufficient to cover the alterations required for functionally adequate but unelaborate site preparation (and air conditioning) costs for a large computer and $50,000 for a medium-scale machine."*

(4) Number of square feet of new office space or building required.

To house people and equipment for program development, office space or additional facilities are required. These additions may cost more than the normal burden rate for flooring space. If it is necessary to establish an entirely new facility at some location, higher costs are encountered because expenses for janitorial work, maintenance, utilities, taxes and so forth are not shared with other ongoing operations.

(5) Exceptional costs of graphic arts and reproduction.

There may be exceptional costs of reproduction during program development, e.g., "polished" brochures and visual aids for training and briefing purposes, profuse illustrations or very large distribution and mailing lists. Useful cost data exist on the expenses associated with various types of reproduction media, but this type of an estimate must be made with regard to each particular new situation.

---

*Controller's Institute Research Foundation, p. 52, op. cit.

(6) Cost of punched cards, magnetic tape and other special supplies or equipment.

Programming efforts normally require the same type of office supplies, stationery, pencils, and so forth, required by other types of desk jobs. In a small automatic data-processing activity this cost of supplies, including magnetic tape reels or punched cards, may be less than one percent. On the other hand, the investment may develop as time goes by. For example, one IBM 7094 facility has a library of approximately 10,000 tapes (2400-foot reels) which cost over $30 per tape--a total of $300,000. This library is incremented by about 400 tapes/year. If the estimator feels that the demands for the particular project will be excessive in this domain in which costs are usually low, a separate estimate for such unusual supplies should be made.

(7) Cost of special security requirements (e.g., Top Secret vault).

A requirement for personnel cleared for Top Secret and for handling highly classified data will undoubtedly add to the total cost. In addition to the cost of obtaining clearances, there is the cost of providing secure work spaces and storage facilities.

III. CONCLUSIONS AND RECOMMENDATIONS

In this document, we try to establish a base for "getting a handle" on the problem of estimating costs for the production of large-scale command and control computer programs. This is an important task because successful project management depends upon an accurate prediction of the resources required to perform the project. Also, this research is a challenging task because it is new, and available literature offers little guidance.

Undoubtedly, more factors were listed here than can ever be integrated into a practical, feasible cost-estimating procedure. In almost all cases, a quantitative relationship of the factor to cost is not known. Further, the factors are very difficult to measure or quantify; they are also highly dependent so that even where they can be quantified, their effect upon other factors must be examined.

Therefore, we consider this listing of cost factors in programming as only a first step toward the development of a more scientific and hopefully a more precise method of estimating the cost of programming efforts. Much more work must be done to determine the significance of each factor and the relationships among factors, and to identify new ones, before a more simplified and reliable cost-estimating relationship can be formulated. This work can be divided into three broad categories:

1. Research and analysis to help define programming activities, skills, and products more rigorously.

2. Data gathering and cost collection specifically related to the processes, activities, and products of computer programming.

3. Experimentation and hypothesis testing to arrive at some conclusion concerning the most cost-effective techniques for implementing computer programming efforts.

Recommendations for work in these areas are given below:

1. Recommendations for Further Analysis. We need to determine whether this list does indeed represent the most significant factors contributing to the cost of computer programs. Also, we need to define much more rigorously the products, skills, and processes with which we are dealing. Examples of specific questions that merit further examination are:

   (a) What is suitable "unit product" in programming? Alternatives might include a block of completely tested instructions of a given size, the entire program system performing a given function, or a certain class of documents.

(b) Can the programming process itself be logically subdivided into clearly recognizable discrete functions with start and finish dates that can be specified?

(c) What exactly do we mean by such terms as program complexity, flexibility, maintainability, etc.?

(d) How are the cost factors related to each other?

(e) How can programming talent or skill be measured to permit comparisons?

2. Recommendations on Cost Collection. The collection of cost data for prediction purposes must be an evolutionary process. Data that are collected will, when analyzed, suggest the framework of the estimating relationships and, by the same token, the theoretical relationships will suggest further data that should be collected. After one iteration, i.e., the collection of some data to suggest the factors listed in this document, we can improve somewhat our suggestions for recording costs in the future. We are cautious because we are aware that recording and collecting data can be expensive, particularly when a definitive plan for data analysis and subsequent use of the results is not available. Nevertheless, we would hope that programming managers would begin to accumulate at least the following information about their projects:

(a) The number of machine language instructions in the program; also, the number of operational and utility program instructions available from other sources at the start of the project and the number of words in tables and the data base.

(b) The number of man months of programmer effort to design, code, test and document the program, including first level of supervision.

(c) The number of hours of machine time required for testing and debugging, and the types of machines used.

(d) The number, types, and timing of important program changes and, in at least a qualitative sense, the effects of these changes on the final product.

(e) The types, number of pages, and format of documentation required.

In addition, it would be useful if a log could be kept by a project "historian" describing certain qualitative attributes such as those identified earlier in this paper. This section should describe the data-processing functions of the program system and its relationship to other program systems and components of the command and control

system. It should identify all interim and end products, and associate them with project schedules. It should also describe the development environment and the management procedures used. Finally such a log should record every change in plans as each affects the costs and schedules and the reasons for each change.

One of the decisions made in preparing these recommendations was the level of effort to be examined and the level for which data would be recorded. One could consider three levels for programming: (1) a total programming contract that may include several different program systems for different purposes, such as operational, utility, and support, and may even include several sequenced versions of these same program systems; (2) an individual program system consisting of a set of subprograms tied together to perform certain functions and associated with a particular operational or delivery date; and (3) an individual program, which might be a subprogram of one of the program systems mentioned above. The decision was to recommend collection of data for program systems, or individual programs distinguishable as the smallest set of computer program instructions (a) whose purpose is defined by someone other than the programmer, (b) that is delivered to the user as a package, and (c) that is loaded into the computer as a program unit or system to achieve the stated objective.

The computer programming products with which to associate cost data are the tapes, listings, and descriptive documents for the components of the program system (e.g., operational, executive, utility programs). Further, cost data can be collected for each of the activities associated with each product (e.g., analysis, design, coding, test, and documentation). Comparison of these data with the orginal estimates will provide "feedback" to estimators and help them understand why estimates deviate from actual costs.

3. Recommendations for Experimentation. There are some factors that can only be analyzed in a controlled environment. The development of more precise definitions recommended in (1) above would provide a basis for examining programming efforts in a pseudo-controlled manner; that is, such definitions would supply descriptive standards. In the absence of these definitions, actual experiments might be conducted. For example, statistically designed experiments could be performed to determine the cost effectiveness of different programming languages, the relation between total cost and computer usage (i.e., turn-around time), and the optimum number of programmers and mix of programming experience for a particular type of program. The chief drawback to experimentation appears to be its cost. The lack of standards that permit comparisons means that an extremely large number of variables must be controlled and, hence, many cases must be examined.

To the extent that it is feasible, we are following these recommendations in the continuing research on cost-estimating relationships. For managers involved in program development, we believe these recommendations are but the beginning of a systematic way of looking at the management of computer programming that will enable them first, to determine costs of the various programming activities more accurately, and then to identify areas in which cost reductions can easily be made.

APPENDIX I

LIST OF COMPUTER PROGRAMMING COST FACTORS

Summarized below for the convenience of the reader is the complete list of cost factors discussed in this paper.

OPERATIONAL REQUIREMENTS AND DESIGN

1. Extent of innovation in the system, its components, and especially the automatic data-processing function.

2. Extent to which the programming designer will participate in a determination of the information-processing needs (i.e., the system and operations analysis, and the system and operational design).

3. Number, size, frequency, and time of system design changes.

4. Extent of system dispersion and number of interfaces.

5. Number of other components and subsystems being developed concurrently as part of the system, e.g., in a command and control system, sensor, and communication subsystems.

PROGRAM DESIGN AND PRODUCTION

1. Number of computer program instructions and the types of programs that must be produced.

2. Number, types, and frequency of inputs and outputs to the computer(s).

3. Extent of innovation required in the program system; that is, the degree to which programs are similar in nature to those previously written.

4. Number, types, and quality of publications and documentation for both customer and internal use.

5. Extent of complexity of the data-processing functions.

6. Degree to which the following program design characteristics are recognized and must be incorporated.

   (a) Maintainability--the ease with which new functions can be detected and corrected.

   (b) Changeability--the ease with which new functions can be incorporated in the program.

(c) Usability--the ease with which personnel other than designers can use the program.

(d) Flexibility--the ease with which the program can be used for other purposes with only slight modification (e.g., SAGE programs for air traffic control).

7. Extent of the constraints on program design.

(a) Computer storage capacity.

(b) Number of input-output channels.

(c) Timing of internal transfers.

(d) Extent to which the program must operate in a real-time mode.

8. Number, size, frequency, and timing of program design changes.

9. Extent to which data for data base are available, or data collection is required.

10. Number of entries (total size) for the data base, the number of different types of data needed for it, and the extent to which each item can serve many programs or subprograms.

11. Efficiency of the programming language and the compiler or assembler.

12. Extent to which programming tools are available and usable.

13. Extent of the completeness and clarity of the system test and acceptance test requirements.

DATA-PROCESSING EQUIPMENT

1. Number of hours per day of computer availability.

2. Extent of capability of the computer and its suitability for the job.

3. Extent to which the operation of the computer and peripheral equipment is reliable, well tested, and well documented.

4. Number of automatic data-processing components being developed concurrently with the program.

5. Number of different computers for which programs are being prepared.

6. Number and types of displays used.

7. Extent to which adequate EAM support will be available.

PROGRAMMING PERSONNEL

1. Types and quality of programmers.

2. Number of man months of programmer training required.

3. Number of programmers to be assigned to a given function or task.

4. Policy of obtaining and phasing of personnel to staff a new program development.

5. Rate of turnover.

MANAGEMENT PROCEDURES

1. Extent of use, maintenance, and monitoring of effective management plans within both the customer's and program developer's organizations.

2. Extent of formalized procedures to use the computer facility.

3. Extent to which there is a well defined and controlled system change procedure.

4. Extent of an error-reporting and -correcting procedure.

5. Extent of contingency plans in the event the computer is overloaded or otherwise unavailable.

6. Extent of quality control that is exercised during testing (e.g., reliability requirements).

DEVELOPMENT ENVIRONMENT

1. Number of agencies with which the programmer contractor must deal and their level of experience with system development.

2. Average number of days and effort required for concurrence.

3. Travel requirements.

4. Extent to which delivery dates for required programming tools are reliable, and correspondingly, the amount of pressure caused by a tight schedule.

5. Extent to which the computer is operated by another agency.

FACILITIES, SERVICES, AND SUPPLIES

1. Number of computer operators and EAM personnel required.
2. Number and experience of technical management personnel, administrative personnel, and technical editors.
3. Cost of special simulation facilities, computer room facilities or special office equipment.
4. Number of square feet of new office space or building required.
5. Exceptional costs of graphic arts and reproduction.
6. Cost of punched cards, magnetic tape and other special supplies or equipment.
7. Cost of special security requirements (e.g., Top Secret vault).

APPENDIX II

CALCULATION OF LEAST SQUARES FIT TO DATA POINTS

y	x		y'	R
810	33.1	$y = a + bx$	544	266
6010	373.4		7928	-1918
2986	71.4	$b = \frac{\Sigma xy - n\bar{x}\bar{y}}{\Sigma x^2 - n\bar{x}^2}$	1375	1611
			2579	-1329
4166	198.6		4135	31
14500	640.0	$a = \bar{y} - b\bar{x}$	13714	786
10400	459.0		9786	614
40122	1902.4			

$\Sigma x = 271.714$

$\Sigma y = 5731.714$

$b = \frac{6,622,029.4}{304,648.2}$

$\Sigma x^2 = 821,447.20$

$\Sigma xy = 17,523,738.0$

$b = 21.7$

$\bar{x}\bar{y} = 1,557,386.938$

$\bar{x}^2 = 73,828.498$

$a = -174.3$

$y' = -174.3 + 21.7x$

$n\bar{x}\bar{y} = 10,901,708.566$

$n\bar{x}^2 = 516,799.486$

y = Number of computer hours (observed)

x = Number of instructions (thousands)

y' = Number of computer hours (estimated)

R = Residual

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

System Development Corporation.
Santa, Monica, Cal.

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

N/A

3. REPORT TITLE

Factors That Affect The Cost of Computer Programming Vol I

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(Last name, first name, initial)*

Farr, L.
Nanus, B.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
Jul 64	59	0

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
AF19(628)1648	TM 1447/000/02
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	ESD-TDR-64-448

10. AVAILABILITY/LIMITATION NOTICES

Qualified Requesters May Obtain Copies From DDC.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Directorate of Computers. ESD L.G. Hanscom Field, Bedford, Mass.

13. ABSTRACT

Although accurate estimation of computer programming costs is an important prerequisite for effective programming management, such estimates have historically been very unreliable. Some of the underlying causes of this problem are discussed, and about fifty factors that appear to contribute to the cost of computer programs are identified. Data concerning the effects of a few of these factors upon cost are presented by way of illustration. Recommendations are made for more detailed cost collection, cost analysis, and experimentation.

DD FORM 1 JAN 64 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Programming Computers Costs Experimental Data Analysis						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through

_____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through

_____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through

_____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.